Use Authorization Mechanisms Correctly

William L. Fithen, Software Engineering Institute [vita[3]]

2005-10-03

Incorrect use of, or failing to use, authorization mechanisms can introduce vulnerability.

Description

The following are frequent authorization design defects that lead to vulnerability:

- Trying to interpret access control rules for lower level subsystems instead of using the subsystems to interpret their rules. This is a common error in setuid programs on UNIX.
- Designing authorization systems with an insufficiently rich privilege menu that encourages privilege overloading. The exemplar for this is the UNIX superuser [POSIX.1e[8], VU#706838[9]].
- *Unauthenticated* authorization systems that appear to control access, but without proper authentication don't control anything [VU#258834[10]].
- Ambiguity of authentication. Many authorization systems use ambiguous symbols (i.e., principal names) to identify principals allowing circumvention of authorization by using a different, though equivalent, principal name. For example, there are many implementations for restricting remote host access to local services that may allow many proper—but apparently different—names for unique hosts (e.g., fully qualified domain names, shortened names, CNAMEs, IPv4 addresses, IPv6 addresses).

Applicable Context

Missing, incomplete, or incorrect application of an authorization mechanism.

Impacts Being Mitigated

- Impact #1:
  - **Minimally:** The least impact of this class of vulnerability is allowing/granting access to a computing resource to a--presumably authentic--individual.
  - **Maximally:** The greatest impact of this class of vulnerability depends on the nature of the resource to which access has been incorrectly granted. In the worse case, the result would be a complete loss of system integrity.

Security Policies to be Preserved

- Policy #1:
  - Access to each computing resource should be granted to only those that have a legitimate

[3] file://portal/vitae/william_l_fithen

[8] #refs

[9] #refs

[10] #refs

requirement for it.

References

[POSIX.1e]	*Draft Standard for Information Technology--Portable Operating System Interface (POSIX)--Part 1: System Application Program Interface (API)--Amendment #: Protection, Audit, and Control Interfaces [C Language]*. IEEE/CS JTC1 22.42. http://wt.xpilot.org/publications/posix.1e/download/Posix_1003. (1997).
[VU#258834]	Gennari, Jeff. *Vulnerability Note VU#258834: WebEOC privileges are based on client-side authorization*. http://www.kb.cert.org/vuls/id/258834 (2005).
[VU#706838]	Dormann, Will. *Vulnerability Note VU#706838: Apple Mac OS X vulnerable to buffer overflow via vpnd daemon*. http://www.kb.cert.org/vuls/id/706838[25] (2005).
[Wright 02]	Wright, Chris; Cowan, Crispin; Morris, James; Smalley, Stephen; & Kroah-Hartman, Greg. *Linux Security Module Framework*. http://lsm.immunix.org/docs/lsm-ols-2002/lsm.pdf (2002).

SEI Copyright

Felder

[25] http://www.kb.cert.org/vuls/id/703838

[1] http://www.sei.cmu.edu/about/legal-permissions.html

Name	Wert
Copyright Holder	SEI

Felder

Name	Wert
is-content-area-overview	false
Content Areas	Knowledge/Guidelines
SDLC Relevance	Implementation
Workflow State	Publishable